

Probleme cu fisiere - Variante BAC SIII, 2009

- V1.** 3. Fișierul text **bac.txt** conține, pe o singură linie, cel mult 1000 de numere naturale nenule cu cel mult 4 cifre fiecare, numerele fiind separate prin câte un spațiu. Scrieți un program C/C++ care citește de la tastatură un număr natural nenul n ($n \leq 999$) și numerele din fișierul **bac.txt** și care afișează pe ecran, separate prin câte un spațiu, toate numerele din fișier care sunt divizibile cu n . Dacă fișierul nu conține niciun astfel de număr, atunci se va afișa pe ecran mesajul **NU EXISTA**.
Exemplu: dacă fișierul **bac.txt** conține numerele: 3 100 40 70 25 5 80 6 3798, pentru $n=10$ atunci pe ecran se va afișa: 100 40 70 80 (10p.)
- V2.** 3. Fișierul text **NR.TXT** conține pe o singură linie, separate prin câte un spațiu, cel mult 100 de numere **întregi**, fiecare număr având cel mult 4 cifre. Scrieți un program C/C++ care citește numerele din fișierul **NR.TXT** și afișează pe ecran, separate prin câte un spațiu, în ordine crescătoare, toate numerele **naturale nenule** din fișier. Dacă nu există astfel de numere se va afișa pe ecran mesajul **NU EXISTA**.
Exemplu: dacă fișierul **NR.TXT** conține numerele: -3 -10 0 7 -5 7 51 -800 6 3798, atunci pe ecran se va afișa: 6 7 7 51 3798 (10p.)
- V3.** 3. Fișierului text **NR.TXT** conține pe o singură linie, separate prin câte un singur spațiu, cel mult 100 de numere naturale, fiecare număr având cel mult 4 cifre. Scrieți un program C/C++ care citește toate numerele din fișierul **NR.TXT** și afișează pe ecran, separate prin câte un spațiu, în ordine crescătoare, toate numerele din fișier care au cel puțin 3 cifre. Dacă fișierul nu conține astfel de numere se va afișa pe ecran mesajul **NU EXISTA**. (10p.)
- V4.** 3. Fișierul text **NR.TXT** conține pe o singură linie, separate prin câte un singur spațiu, cel mult 100 de numere naturale, fiecare număr având cel mult 4 cifre. Scrieți un program C/C++ care citește numerele din fișierul **NR.TXT** și afișează pe ecran, separate prin câte un spațiu, în ordine descrescătoare, toate numerele din fișier care au cel mult 2 cifre. Dacă fișierul nu conține astfel de numere se va afișa pe ecran mesajul **NU EXISTA**. (10p.)
- V5.** 3. Scrieți un program C/C++ care citește de la tastatură un număr natural n cu cel mult 8 cifre ($n \geq 10$) și care creează fișierul text **NR.TXT** ce conține numărul n și toate prefixele nenule ale acestuia, pe o singură linie, separate prin câte un spațiu, în ordine descrescătoare a valorii lor.
Exemplu: pentru $n=10305$ fișierul **NR.TXT** va conține numerele:
10305 1030 103 10 1 (10p.)
- V6.** 4. Se consideră fișierul **BAC.TXT** ce conține un șir **crescător** cu cel mult un milion de numere naturale de cel mult nouă cifre fiecare, separate prin câte un spațiu.
a) Să se scrie un program C/C++ care, folosind un algoritm eficient din punct de vedere al memoriei utilizate și al timpului de executare, citește din fișier toți termenii șirului și afișează pe ecran, pe o singură linie, fiecare termen distinct al șirului urmat de numărul de apariții ale acestuia în șir. Valorile afișate sunt separate prin câte un spațiu.
Exemplu: dacă fișierul **BAC.TXT** are următorul conținut:
1 1 1 5 5 5 5 9 9 11 20 20 20
programul va afișa:
1 3 5 4 9 2 11 1 20 3
deoarece 1 apare de 3 ori, 5 apare de 4 ori, etc. (6p.)
b) Descrieți succint, în limbaj natural, metoda de rezolvare folosită, explicând în ce constă eficiența ei (3 – 4 rânduri). (4p.)

- V7. 3. Scrieți un program C/C++ care citește de la tastatură un număr natural n ($0 < n \leq 100$) și cele $3 \cdot n$ elemente ale tabloului unidimensional v , fiecare element fiind un număr natural cu cel mult patru cifre fiecare. Tabloul este împărțit în trei zone, cu câte n elemente: prima zonă conține primele n elemente din tablou, a doua zonă conține următoarele n elemente din tablou, restul elementelor fiind în zona a treia. Programul va interschimba primul element par (dacă există) al zonei unu cu ultimul element impar (dacă există) al zonei trei și apoi va scrie pe prima linie a fișierului text **BAC.TXT** toate elementele tabloului, separate prin câte un spațiu. În cazul în care unul dintre aceste două elemente, care urmează a fi interschimbate, nu există, programul nu va efectua nici o modificare asupra tabloului dat.
Exemplu: pentru $n=3$ și $v=(1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9)$, fișierul **BAC.TXT** va conține:
 1 9 3 4 5 6 7 8 2 (10p.)
- V8. 3. Pe prima linie a fișierului text **BAC.TXT** se află o valoare naturală n ($1 < n \leq 50$), iar pe a doua linie n numere naturale cu maximum 4 cifre fiecare, despărțite prin câte un spațiu. În șirul numerelor de pe a doua linie a fișierului există cel puțin două numere pătrate perfecte. Scrieți un program C/C++ care citește toate numerele din fișier și afișează pe ecran expresia aritmetică reprezentând suma numerelor de pe a doua linie a fișierului care au proprietatea că sunt pătrate perfecte, cu simbolul + între ele și, după un semn =, valoarea acestei sume, ca în exemplu. Termenii sumei afișate se pot afla în orice ordine.
Exemplu: dacă fișierul **BAC.TXT** are următorul conținut
 5
 9 5 36 9 8
 atunci pe ecran se poate afișa:
 9+9+36=54 sau 9+36+9=54 sau 36+9+9=54 (10p.)
- V9. 4. Se consideră fișierul **BAC.TXT** ce conține cel mult un milion de numere naturale separate prin spații, fiecare număr având cel mult nouă cifre.
 a) Scrieți un program C/C++ care citește toate numerele din fișierul **BAC.TXT** și determină, folosind un algoritm eficient din punct de vedere timpului de executare, cele mai mari două numere de trei cifre care nu se află în fișier. Cele două numere vor fi afișate pe ecran în ordine descrescătoare, cu un spațiu între ele. Dacă nu pot fi determinate două astfel de numere, programul va afișa pe ecran valoarea 0.
Exemplu: dacă fișierul **BAC.TXT** conține numerele:
 12 2345 123 67 989 6 999 123 67 989 999
 atunci programul va afișa
 998 997 (6p.)
 b) Descrieți succint, în limbaj natural, metoda de rezolvare folosită, explicând în ce constă eficiența ei (3 – 4 rânduri). (4p.)

4. Evidența produselor vândute de o societate comercială este păstrată în fișierul **PRODUSE.TXT**. Pentru fiecare vânzare se cunosc: tipul produsului (un număr natural de cel mult 4 cifre), cantitatea vândută exprimată în kilograme (un număr natural mai mic sau egal cu 100) și prețul unui kilogram (un număr natural mai mic sau egal cu 100).

V10.

Fișierul **PRODUSE.TXT** are cel mult 200000 de linii și fiecare linie conține trei numere naturale, separate prin câte un spațiu, ce reprezintă, în această ordine tipul, cantitatea și prețul de vânzare al unui produs la momentul vânzării respective.

a) Să se scrie un program C/C++ care, utilizând un algoritm eficient din punct de vedere al timpului de executare, determină pentru fiecare tip de produs vândut suma totală obținută în urma vânzărilor. Programul va afișa pe câte o linie a ecranului tipul produsului și suma totală obținută, separate prin câte un spațiu, ca în exemplu.

Exemplu: dacă fișierul **PRODUSE.TXT** are conținutul alăturat, programul va afișa perechile următoare, nu neapărat în această ordine:

1 150

2 30

3 5

3 1 5
1 20 5
2 10 3
1 10 5

(6p.)

b) Descrieți succint, în limbaj natural, metoda de rezolvare folosită, explicând în ce constă eficiența (3 - 4 rânduri).

(4p.)

3. Fișierul text **numere.txt** conține pe prima linie un număr natural n ($n < 30000$), iar pe a doua linie n numere întregi având maximum 4 cifre fiecare. Se cere să se afișeze pe ecran un șir de n numere întregi, cu proprietatea că valoarea termenului de pe poziția i ($i=1, 2, \dots, n$) din acest șir este egală cu cea mai mare dintre primele i valori de pe a doua linie a fișierului **numere.txt**.

V11.

a) Descrieți pe scurt un algoritm de rezolvare, eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat, explicând în ce constă eficiența sa.

(4p.)

b) Scrieți programul C/C++ corespunzător algoritmului descris.

(6p.)

Exemplu: dacă fișierul **numere.txt** are conținutul

alăturat, se afișează pe ecran numerele

4 6 6 7 8 8 8 8 9 10 10

12
4 6 3 7 8 1 6 2 7 9 10 8

3. Fișierele text **NR1.TXT** și **NR2.TXT** conțin, separate prin câte un spațiu, mai multe numere întregi de cel mult 9 cifre fiecare. Fiecare dintre fișiere conține cel mult 100 de valori și numerele din fiecare fișier sunt ordonate strict crescător. Se cere să se afișeze pe ecran, în ordine crescătoare, numerele divizibile cu 5 care se găsesc doar în unul din cele două fișiere.

V12.

Exemplu: dacă fișierul **NR1.TXT** conține numerele 1 2 3 4 7 20 60, iar fișierul **NR2.TXT** conține numerele 3 5 7 8 9 10 12 20 24, atunci se vor afișa pe ecran valorile 5 10 60.

a) Descrieți un algoritm de rezolvare a acestei probleme, eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat, explicând în ce constă eficiența acestuia.

(4p.)

b) Scrieți programul C/C++ corespunzător algoritmului descris.

(6p.)

- V13.** 4. Se consideră subprogramul **P** care are doi parametri:
 – **n**, prin intermediul căruia primește un număr natural de cel mult 9 cifre
 – **c**, prin intermediul căruia primește o cifră.
 Subprogramul va furniza tot prin intermediul parametrului **n** numărul obținut din **n** prin eliminarea tuturor aparițiilor cifrei **c**. Dacă, după eliminare, numărul nu mai conține nicio cifră sau conține doar cifre 0, rezultatul returnat va fi 0.
- a)** Scrieți doar antetul subprogramului **P**. **(2p.)**
- b)** Pe prima linie a fișierului text **BAC.IN** se găsesc, separate prin câte un spațiu, mai multe numere naturale de cel mult 9 cifre fiecare. Scrieți programul **C/C++** care citește numerele din acest fișier, utilizând apeluri ale subprogramului **P** elimină toate cifrele impare din fiecare dintre aceste numere și apoi scrie în fișierul text **BAC.OUT** numerele astfel obținute, separate prin câte un spațiu. Dacă un număr din fișierul **BAC.IN** nu conține nicio cifră pară nenulă, acesta nu va mai apărea deloc în fișierul de ieșire. **(8p.)**
- Exemplu:** dacă fișierul **BAC.IN** conține numerele 25 7 38 1030 45127 0 35 60 15, atunci **BAC.OUT** va avea conținutul: 2 8 42 60.
- V14.** 4. Fișierul text **BAC.TXT** conține mai multe numere naturale, cu cel mult 6 cifre fiecare, câte un număr pe fiecare linie a fișierului.
 Scrieți un program **C/C++** care citește toate numerele din fișierul **BAC.TXT** și le afișează pe ecran, în aceeași ordine, câte **cinci** pe fiecare linie, separate prin câte un spațiu, cu excepția ultimei linii care poate conține mai puțin de cinci numere. Programul va afișa apoi pe ecran, pe o linie separată, câte numere din fișier au suma cifrelor pară.
- Exemplu:** dacă fișierul are conținutul alăturat, pe ecran se vor afișa numerele de mai jos:
- | | |
|----|----|
| 11 | 11 |
| 21 | 21 |
| 30 | 30 |
| 40 | 40 |
| 51 | 51 |
| 16 | 16 |
| 17 | 17 |
| 10 | 10 |
| 1 | 1 |
- 4 **(10p.)**
- V15.** 4. În fișierul text **BAC.IN** se găsesc, pe o singură linie, separate prin câte un spațiu, mai multe numere naturale de cel mult 6 cifre fiecare. Se cere să se determine și să se afișeze pe ecran, separate printr-un spațiu, ultimele **două** numere impare (nu neapărat distincte) din fișierul **BAC.IN**. Dacă în fișier se găsește un singur număr impar sau niciun număr impar se va scrie pe ecran mesajul **Numere insuficiente**.
- Exemplu:** dacă fișierul **BAC.IN** conține valorile: 12 15 68 13 17 90 31 42 se va afișa 17 31.
- a)** Descrieți în limbaj natural un algoritm eficient din punct de vedere al spațiului de memorie și al timpului de executare, pentru rezolvarea acestei probleme, explicând în ce constă eficiența acestuia. **(4p.)**
- b)** Scrieți programul **C/C++** corespunzător algoritmului descris. **(6p.)**

V16.

4. În fișierul `numere.txt` sunt memorate maximum 10000 de numere naturale cu cel mult 9 cifre fiecare. Fiecare linie a fișierului conține câte un număr. Se cere afișarea pe ecran, în ordine descrescătoare, a tuturor cifrelor care apar în numerele din fișier. Alegeți un algoritm de rezolvare eficient din punct de vedere al timpului de executare.

Exemplu: dacă fișierul `numere.txt` conține:

267

39628

79

se va tipări 9987766322.

a) Descrieți succint, în limbaj natural, strategia de rezolvare și justificați eficiența algoritmului ales. (4p.)

b) Scrieți programul C/C++ corespunzător algoritmului ales. (6p.)

V17.

4. În fișierul `numere.txt` pe prima linie este memorat un număr natural n ($n \leq 10000$), iar pe linia următoare un șir de n numere naturale distincte două câte două, separate prin câte un spațiu, cu maximum 4 cifre fiecare. Se cere afișarea pe ecran a poziției pe care s-ar găsi primul element din șirul aflat pe linia a doua a fișierului, în cazul în care șirul ar fi ordonat crescător. Numerotarea pozițiilor elementelor în cadrul șirului este de la 1 la n . Alegeți un algoritm de rezolvare eficient din punct de vedere al memoriei utilizate și al timpului de executare.

Exemplu: dacă fișierul `numere.txt` conține:

6

267 13 45 628 7 79

se va afișa 5, deoarece primul element din șirul inițial, 267, s-ar găsi pe poziția a cincea în șirul ordonat crescător (7 13 45 79 267 628).

a) Descrieți succint, în limbaj natural, strategia de rezolvare și justificați eficiența algoritmului ales. (4p.)

b) Scrieți programul C/C++ corespunzător algoritmului ales. (6p.)

V18.

4. În fișierul `numere.txt` este memorat un șir de maximum 10000 numere naturale, distincte două câte două, cu maximum 4 cifre fiecare, separate prin câte un spațiu. Pentru un număr k citit de la tastatură, se cere afișarea pe ecran a poziției pe care se va găsi acesta în șirul de numere din fișier, dacă șirul ar fi ordonat descrescător, sau mesajul **nu există**, dacă numărul k nu se află printre numerele din fișier. Alegeți un algoritm eficient de rezolvare din punct de vedere al memoriei utilizate și al timpului de executare.

Exemplu: dacă fișierul `numere.txt` conține numerele 26 2 5 30 13 45 62 7 79, iar k are valoarea 13, se va afișa 6 deoarece 13 s-ar găsi pe poziția a șasea în șirul ordonat descrescător (79 62 45 30 26 13 7 5 2).

a) Descrieți succint, în limbaj natural, strategia de rezolvare și justificați eficiența algoritmului ales. (4p.)

b) Scrieți programul C/C++ corespunzător algoritmului ales. (6p.)

V19.

4. În fișierul **nr1.txt** este memorată pe prima linie o valoare naturală **n** de cel mult 8 cifre, iar pe linia următoare sunt memorate **n** numere naturale, cu maximum 4 cifre fiecare, ordonate strict crescător și separate prin câte un spațiu. În fișierul **nr2.txt** este memorată pe prima linie o valoare naturală **m** de cel mult 8 cifre, iar pe linia următoare sunt memorate **m** numere naturale, cu maximum 4 cifre fiecare, ordonate strict crescător și separate prin câte un spațiu. Se cere afișarea pe ecran, separate prin câte un spațiu, în ordine strict crescătoare, a tuturor numerelor aflate pe a doua linie în cel puțin unul dintre cele două fișiere. În cazul în care un număr apare în ambele fișiere, el va fi afișat o singură dată. Alegeți un algoritm de rezolvare eficient din punct de vedere al memoriei utilizate și al timpului de executare.

Exemplu: pentru următoarele fișiere:

nr1.txt

5

3 6 8 9 12

se va afișa 2 3 5 6 7 8 9 12 13.

nr2.txt

6

2 3 5 7 9 13

a) Descrieți succint, în limbaj natural, strategia de rezolvare și justificați eficiența algoritmului ales. (4p.)

b) Scrieți programul C/C++ corespunzător algoritmului ales. (6p.)

V20.

4. În fișierul **nr1.txt** este memorată pe prima linie o valoare naturală **n** de cel mult 8 cifre, iar pe linia următoare sunt memorate **n** numere naturale, cu maximum 4 cifre fiecare, ordonate strict crescător și separate prin câte un spațiu. În fișierul **nr2.txt** este memorată pe prima linie o valoare naturală **m** de cel mult 8 cifre, iar pe linia următoare sunt memorate **m** numere naturale, cu maximum 4 cifre fiecare, ordonate strict crescător și separate prin câte un spațiu. Se cere afișarea pe ecran, separate prin câte un spațiu, în ordine strict crescătoare, a tuturor numerelor aflate pe a doua linie atât în primul cât și în al doilea fișier. Alegeți un algoritm de rezolvare eficient din punct de vedere al memoriei utilizate și al timpului de executare.

Exemplu: pentru următoarele fișiere:

nr1.txt

5

3 6 8 9 12

se va afișa 3 9.

nr2.txt

6

2 3 5 7 9 13

a) Descrieți succint, în limbaj natural, strategia de rezolvare și justificați eficiența algoritmului ales. (4p.)

b) Scrieți programul C/C++ corespunzător algoritmului ales. (6p.)

4. Fișierul text **BAC.TXT** conține pe prima linie două numere naturale n și k separate de un spațiu ($3 \leq n \leq 10000$, $2 \leq k \leq n/2$), iar pe a doua linie un șir de n numere naturale x_1, x_2, \dots, x_n separate prin câte un spațiu, fiecare număr din acest șir având cel mult patru cifre.

V21.

a) Scrieți un program C/C++ care citește numerele din fișier și determină, utilizând o metodă eficientă din punct de vedere al timpului de executare, cel mai mic indice i ($1 \leq i \leq n-k+1$) pentru care media aritmetică a numerelor $x_i, x_{i+1}, \dots, x_{i+k-1}$ este maximă. Programul afișează valoarea lui i pe ecran.

Exemplu: pentru fișierul alăturat se afișează 2, deoarece media maximă se obține pentru 9, 4, 7.

(6p.)

8	3
2	9 4 7 5 2 9 9

b) Explicați succint, în limbaj natural, metoda utilizată la punctul a, justificând eficiența acesteia. (4p.)

4. Scrieți programul C/C++ care citește din fișierul text **BAC.TXT** numărul întreg n ($1 \leq n \leq 10000$) și un șir de n perechi de numere întregi a, b ($1 \leq a \leq b \leq 32000$), fiecare pereche fiind scrisă pe o linie nouă a fișierului, cu un spațiu între cele două numere. Programul afișează pe ecran pentru fiecare pereche a, b cel mai mare număr natural din intervalul închis $[a, b]$ care este o putere a lui 2 sau numărul 0 dacă nu există nicio putere a lui 2 în intervalul respectiv. Numerele afișate pe ecran se scriu în linie, separate prin câte un spațiu. Un număr p este putere a lui 2 dacă există un număr natural k astfel încât $p=2^k$.

Exemplu: dacă fișierul **BAC.TXT** conține numerele

3
2 69
10 20
19 25

se va afișa: 64 16 0.

(10p.)

4. Fișierul text **BAC.TXT** conține pe prima linie un număr natural nenul n ($1 \leq n \leq 1000$), iar pe fiecare dintre următoarele n linii, câte două numere întregi a și b ($1 \leq a \leq b \leq 32000$), fiecare pereche reprezentând un interval închis de forma $[a, b]$. Scrieți un program C/C++ care determină intervalele care au proprietatea că intersecția cu oricare dintre celelalte $n-1$ intervale este vidă și afișează pe câte o linie a ecranului, separate printr-un spațiu, numerele care reprezintă capetele intervalelor determinate. Dacă nu există nici un astfel de interval, se afișează pe ecran mesajul **NU EXISTA**. (10p.)

Exemplu: dacă fișierul **BAC.TXT** are conținutul alăturat, pe ecran se va afișa:

2 6 sau 17 20
17 20 2 6

4
17 20
2 6
10 15
8 16

V23.

- V24.**
4. Fișierul text **bac.txt** conține pe prima linie numărul natural n , $1 \leq n \leq 30000$, pe următoarele n linii un șir de n numere întregi, ordonate crescător, iar pe ultima linie două numere întregi a și b ($a \leq b$) separate de un spațiu. Fiecare dintre cele n numere, precum și valorile a și b , au cel mult patru cifre.
- a) Scrieți un program C/C++, eficient din punct de vedere al timpului de executare, care afișează pe ecran cel mai mic număr întreg din intervalul închis $[a, b]$ care se găsește în șirul dat. Dacă nu există un astfel de număr, programul afișează textul **NU**.
Exemplu: dacă fișierul **bac.txt** are conținutul alăturat, programul afișează
- | | |
|------|------|
| 4 | 4 |
| -2 | -2 |
| 7 | 7 |
| 11 | 11 |
| 35 | 35 |
| 8 15 | 8 15 |
- (6p.)**
- b) Descrieți în limbaj natural metoda utilizată și explicați în ce constă eficiența ei.
- (4p.)**
- V25.**
4. Fișierul text **NUMAR.TXT** conține pe prima linie un număr real pozitiv x care are cel mult două cifre la partea întreagă și cel mult șapte cifre după punctul zecimal..
- a) Scrieți un program C/C++ care, utilizând un algoritm eficient din punct de vedere al timpului de executare și al memoriei utilizate, afișează pe ecran, separate printr-un spațiu, două numere naturale al căror raport este egal cu x și a căror diferență absolută este minimă.
Exemplu: dacă fișierul conține valoarea alăturată, se vor afișa pe ecran
- | | |
|-------|-------|
| 0.375 | 0.375 |
|-------|-------|
- (6p.)**
- b) Descrieți în limbaj natural metoda utilizată și explicați în ce constă eficiența ei. **(4p.)**
- V26.**
4. a) Scrieți definiția completă a subprogramului **sterge**, care primește prin cei 4 parametri v, n, i, j :
- v , un tablou unidimensional cu maximum 100 de elemente întregi din intervalul $[-1000, 1000]$
 - n , un număr natural reprezentând numărul de elemente din tabloul v
 - i și j două valori naturale cu $1 \leq i \leq j \leq n$
- și elimină din tabloul v elementele v_i, v_{i+1}, \dots, v_j actualizând valoarea parametrului n .
 Tabloul modificat este furnizat tot prin parametrul v . **(6p.)**
- b) Fișierul text **NUMERE.IN** conține pe prima linie un număr natural nenul n ($1 \leq n \leq 100$) și pe următoarea linie n numere întregi din intervalul $[-1000, 1000]$, separate prin câte un spațiu. Scrieți un program C/C++ care citește din fișierul **NUMERE.IN** numărul natural n , construiește în memorie un tablou unidimensional v cu cele n numere întregi aflate pe linia a doua în fișier și utilizează apeluri utile ale subprogramului **sterge** pentru a elimina din tablou un număr minim de elemente astfel încât să nu existe două elemente alăturate cu aceeași valoare. Elementele tabloului obținut se afișează pe ecran, separate prin câte un spațiu.
Exemplu: Dacă fișierul **NUMERE.IN** are conținutul:
- 12
- 10 10 2 2 19 9 9 9 9 15 15 15 atunci se afișează 10 2 19 9 15. **(10p.)**

4. Fișierul text **NUMERE.IN** conține pe prima linie un număr natural nenul n ($2 \leq n \leq 100$) și pe următoarea linie n numere reale pozitive, în ordine strict crescătoare, separate prin câte un spațiu.

V27.

a) Scrieți un program C/C++ care, utilizând un algoritm eficient din punct de vedere al memoriei utilizate, determină și afișează pe ecran cel mai mare număr natural x cu proprietatea că în orice interval deschis având drept capete oricare două dintre cele n numere aflate pe linia a doua în fișierul **NUMERE.IN** se găsesc cel puțin x numere întregi.

Exemplu: dacă fișierul **NUMERE.IN** are conținutul:

```
6
3.5  5.1  9.2  16  20.33 100
atunci se afișează 2
```

Explicație: în oricare dintre intervalele $(3.5, 5.1)$, $(3.5, 9.2)$, $(3.5, 16)$, $(3.5, 20.33)$, $(3.5, 100)$, $(5.1, 9.2)$, $(5.1, 16)$, $(5.1, 20.33)$, $(5.1, 100)$, $(9.2, 16)$, $(9.2, 20.33)$, $(9.2, 100)$, $(16, 20.33)$, $(16, 100)$, $(20, 33, 100)$ există cel puțin două numere întregi.

b) Descrieți în limbaj natural metoda utilizată și explicați în ce constă eficiența ei. **(4p.)**

4. a) Scrieți definiția completă a unui subprogram **primul**, care
- primește prin singurul său parametru, a , o valoare naturală din intervalul $[2, 10000]$
- returnează o valoare naturală reprezentând cel mai mic divizor al numărului a mai mare strict decât 1. **(6p.)**

V28.

b) Fișierul text **NUMERE.IN** conține pe prima linie un număr natural nenul n ($1 \leq n \leq 100$) și pe următoarea linie n numere naturale din intervalul $[2, 10000]$ separate prin câte un spațiu.

Un număr natural n se numește „**aproape prim**” dacă este egal cu produsul a două numere prime distincte. De exemplu, numărul 14 este „aproape prim” pentru că este egal cu produsul numerelor prime 2 și 7.

Scrieți un program C/C++ care determină, folosind apeluri utile ale suprogramului **primul**, cel mai mare număr „**aproape prim**” de pe linia a doua a fișierului **NUMERE.IN**. În cazul în care există un astfel de număr se afișează pe ecran mesajul **DA**, urmat de numărul determinat, iar în caz contrar mesajul **NU**.

Exemplu: dacă fișierul **NUMERE.IN** are conținutul:

```
6
100 14 21 8 77 35
atunci se afișează pe ecran DA 77 pentru că numărul 77 este cel mai mare dintre numerele
„aproape prime” din fișier (14=7*2, 21=7*3, 77=7*11, 35=7*5). (10p.)
```

V29.

4. Se consideră două tablouri unidimensionale **A** și **B** cu elemente numere naturale din intervalul $[1, 10000]$. Spunem că tabloul **A** “**se poate reduce**” la tabloul **B** dacă există o împărțire a tabloului **A** în secvențe disjuncte de elemente aflate pe poziții consecutive în tabloul **A** astfel încât prin înlocuirea secvențelor cu suma elementelor din secvență să se obțină, în ordine, elementele tabloului **B**.

De exemplu tabloul

A	7	3	4	1	6	4	6	9	7	1	8	7
B	14			7		26				16		

se poate reduce la tabloul

Fișierul text **NUMERE.IN** conține pe prima linie două numere naturale nenule n și m ($1 \leq m \leq n \leq 100$), pe linia a doua n numere naturale din intervalul $[1, 10000]$ și pe linia a treia alte m numere naturale din intervalul $[1, 10000]$. Pe fiecare linie numerele sunt separate prin câte un spațiu.

a) Scrieți un program C/C++ care citește toate numerele din fișierul **NUMERE.IN** și verifică, utilizând un algoritm eficient din punctul de vedere al timpului de executare, dacă tabloul construit cu cele n numere aflate pe linia a doua în fișier se poate reduce la tabloul construit cu cele m numere aflate pe linia a treia în fișier. Programul afișează pe ecran mesajul **DA** în caz afirmativ și mesajul **NU** în caz negativ. (6p.)

b) Descrieți în limbaj natural metoda utilizată și explicați în ce constă eficiența ei. (4p.)

4. Fișierul text **NUMERE.IN** conține pe prima linie un număr natural nenul n ($1 \leq n \leq 100$) și pe următoarea linie n numere reale pozitive **ordonate crescător**, separate prin câte un spațiu.

V30.

a) Scrieți un program C/C++ care citește din fișierul **NUMERE.IN** numărul natural n , și determină, utilizând un algoritm eficient din punct de vedere al timpului de executare și al memoriei utilizate, numărul **minim** de intervale închise de forma $[x, x+1]$, cu x număr natural, a căror reuniune include toate numerele reale din fișier.

Exemplu: Dacă fișierul **NUMERE.IN** are conținutul:

6

2.3 2.3 2.8 5.7 5.7 6.3

atunci se afișează 3 (intervalele $[2, 3]$, $[5, 6]$, $[6, 7]$ sunt cele 3 intervale de forma cerută care conțin numere din șir). (6p.)

b) Descrieți în limbaj natural metoda utilizată și explicați în ce constă eficiența ei. (4p.)

4. În fișierul **numere.txt** se află memorate, pe prima linie un număr natural n ($1 \leq n \leq 100$), iar pe fiecare dintre următoarele n linii, câte două numere întregi x, y ($-100 \leq x \leq y \leq 100$), reprezentând capetele câte unui segment $[x, y]$ desenat pe axa ox de coordonate.

V31.

a) Scrieți în limbajul C/C++ un program eficient din punct de vedere al timpului de executare și al spațiului de memorare, care citește din fișier datele existente, determină segmentul rezultat în urma intersecției tuturor celor n segmente date și afișează pe ecran două numere despărțite printr-un spațiu ce reprezintă capetele segmentului cerut. Dacă segmentele nu au nici un punct comun se va afișa pe ecran valoarea 0. (6p.)

b) Descrieți în limbaj natural algoritmul utilizat, justificând eficiența acestuia. (4p.)

Exemplu: dacă fișierul **numere.txt** are conținutul alăturat, se va afișa

pe ecran	5
3 5	-7 10
	3 20
	-5 5
	0 12
	-8 30

4. În fișierul **numere.txt** sunt memorate pe mai multe linii, numere întregi (cel mult 100), numerele de pe aceeași linie fiind despărțite prin câte un spațiu, fiecare număr având cel mult 9 cifre. Să se determine cele mai mici două valori distincte, fiecare având **exact** două cifre, memorate în fișier și să se afișeze pe ecran aceste valori, despărțite printr-un spațiu. Dacă în fișier nu se află două astfel de valori, pe ecran se va afișa valoarea 0.

V32.

a) Descrieți în limbaj natural o metodă de rezolvare eficientă din punct de vedere al gestionării memoriei și al timpului de executare. (4p.)

b) Scrieți programul C/C++ corespunzător metodei descrise la punctul a. (6p.)

Exemplu: dacă fișierul **numere.txt** are conținutul alăturat, se

va afișa pe ecran, nu neapărat în această ordine:	5 10
-77 10	3 -77 20
	50 5 0 12 18 30

4. Pe prima linie a fișierului **numere.txt** se află un număr natural n ($n \leq 100$), iar pe următoarele n linii, câte n numere naturale despărțite prin câte un spațiu, fiecare având cel mult 9 cifre. Printre aceste numere se află cel puțin unul cu 3 cifre și cel puțin unul cu 4 cifre.

V33.

a) Scrieți în limbajul C/C++, un algoritm eficient din punct de vedere al gestionării memoriei care citește din fișier datele existente și determină și afișează pe ecran, separate printr-un spațiu, două numere din fișier, x și y , unde x este cel mai mare număr de trei cifre, iar y este acel număr pentru care $|x - y|$ are valoare minimă. Dacă sunt mai multe valori pentru y care respectă condiția impusă se va afișa numai una dintre ele. (10p.)

b) Explicați în limbaj natural metoda utilizată justificând eficiența acesteia. (4p.)

Exemplu: dacă fișierul **numere.txt** are

conținutul alăturat, se va afișa:	5
800 1100	112 333 1 500 1100
	1 95 7 97 12
	45 800 0 7 89
	1 5 17 197 102
	45 86 0 7 9